A Coherent Modularity Driven Framework for Global Software Development

Raj Gopal

Roll No. 213CS3182

under the supervision of

Prof. Durga Prasad Mohapatra

Department of C.S.E

NIT Rourkela

Rourkela – 769008, India

A Coherent Modularity Driven Framework for Global Software Development

Dissertation submitted on

May 2015

to the department of

C.S.E

of

NIT, Rourkela

in partial fulfillment of the requirements

for the degree of

Master of Technology

by

Raj Gopal

(Roll. 213CS3182)

under the supervision of

Prof. Durga Prasad Mohapatra

Department of C.S.E

NIT Rourkela

Rourkela – 769 008, India

Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela-769 008, India. www.nitrkl.ac.in

May 23, 2015

Certificate

This is to certify that the work in the thesis entitled *A Coherent Modularity Driven Framework for Global Software Development* by *Raj Gopal*, having roll number 213CS3182, is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Master of Technology* in *Computer Science and Engineering Department*. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Dr. Durga Prasad Mohapatra

Associate Professor

Department of CSE

NIT, Rourkela

Acknowledgment

First of all, i want to pass on my profound feeling of appreciation and admiration for my supervisor Prof. Durga Prasad Mohapatra, who has been pushing me for my work. I desire to express gratitude to him for acquainting me with the area of Modularization and Global Software Development and giving me the opportunity to work under him. Without his precious guidance and help it would have never been feasible for me to complete the thesis. I am extremely obliged to him for his regular support and guidance in every part of my thesis. I think of it as my favorable luck to have got a chance to work with such a superb person.

I thank my H.O.D. Prof. Santanu Kumar Rath for his consistent backing in my thesis work. He has been huge source of motivation to me and I say thanks to him from my inner heart.

Similarly, i want to thank all employees, PhD researchers, my seniors and youngsters and all associates to give me their general proposals and consolations between the entire work.

I like to thank all staff members and secretarial staff of the CSE Department for their sympathetic cooperation.

Finally last but not the least I am very thankful to my family who helped me in all my troubles and whenever i needed them. They also helped me to battle for me routinely during my troublesome times.

Raj Gopal

Abstract

An appropriate design is required to reduce the problems associated with the software development and maximize the benefits. Modularization in the other way can help in improving the design which can also help in Global Software Development(GSD). In GSD, Modularization can help in distributing the work in different regions. Work is distributed after a thorough study of different tasks and various location, So to do this, categorization of work should be undertaken before it is distributed. This thesis is overall divided into two parts. The first part helps in finding modules using two different analysis. The second part uses the final output of the first part to find coupling and cohesion. In globally distributed software projects,it is possible to have a discrepancy between organizational structure and software architecture which may diminish the overall production. The final aim of software modularization is to increase software project quality and productivity. This thesis is subdivided into two different parts. The first part helps in finding modules using two different analysis : Dependency and Functionality Analysis.

In dependency analysis the dependency among modules are checked and the best modular structure is found of it and in functionality analysis the components having functions together are taken to form the best modular structure and after this the best modular structure is found out using both the analysis. In the second part, coupling and cohesion are found using information theory approach .

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, Softwares is developed globally all over the globe, which calls for multiple teams to complete the project and deliver the software on time. As the software is globally developed the distribution of work should be more important as the production of one workplace could be the input of other work. The thesis is split into two sections. First part explains how we find the best module from the task. In this part we further subdivide into two different ways to discover the best module. They are **Dependency Analysis** and **Functionality Analysis**. In the second section we find coupling and cohesion of the best module found out in the beginning part. To find coupling and cohesion we are using Information theory approach.

1.1 Motivation

Large software projects are developed by multiple teams to meet deadlines and deliver the product on time. In a Global Software Development (GSD) environment, multiple teams are placed in geographically dispersed locations and the work distribution process has become an important function in GSD [1]. The work distribution process can assists in benefiting GSD(e.g. availability of people, closeness to the customer) and its risks(e.g. communication problem, Inexperienced workforce) [2]. Hence, employment should be dealt out after a thorough work on the

different projects and the diverse localizations. to do this, categorization of work should be undertaken before it is distributed.

The notion of modularity is key to the purpose and production of software artifacts, especially for big and complex tasks. In globally distributed software projects, it is definite to experience a mismatch between organizational structure and software architecture which may decrease production. The final aim of software modularization is to increase software quality and productivity [3]. Moreover, properly modularized software is easy to maintain and is of great assistance to the developer [9].

1.2 Objectives

The main objectives of the research work undertaken are:

- To find out a suitable module structure using the dependency of the components in a component diagram.
- To find out a suitable module structure using the functionality of the components in a component diagram.
- To find out the best module structure using the modules found out from dependency and functionality analysis.
- To find out the coupling and cohesion among the identified modules.

1.3 Thesis Organization

The thesis is organised as follow :

1. Chapter 1: In this chapter, a brief introduction about Global Software Development , motivation and objective was discussed.

2. Chapter 2: In the second chapter, some basic concepts and definitions of modularization, Global Software Development, modular system, coupling and cohesion is explained which helps in understanding the theoretical concepts further.

3. Chapter 3: In the third chapter, literature review of all the papers used for this thesis is present which helps in refering to these papers.

4. Chapter 4: In the fourth chapter, concept of modularization is discussed. It deals with the concept of how the best modules are formed using different analysis.

5. Chapter 5: In the fifth chapter, concept of coupling and cohesion of modular system is discussed.

6. Chapter 6: In the sixth chapter, overall conclusion of the thesis is discussed.

Chapter 2

BASIC DEFINITIONS AND CONCEPTS

2.1 Background Details

2.1.1 Global Software Development (GSD)

Global Software Development (GSD) [1] is becoming a measure in the software industry and it is becoming more difficult to implement these effective strategies. Normally, large software projects are produced by multiple teams to see deadlines and present the product on time. In GSD environment, multiple teams are placed in geographically dispersed locations and the work distribution process is an important function in GSD. The work distribution process can assist in benefiting GSD (e.g. Price reduction, availability of people) and its perils (e.g. Inexperienced workforce, communication overhead).

Employment is administered after a thoroughgoing survey of different jobs and various locations so to do this categorization of work should be tackled before it is disseminated. This is what modularization is. The idea of modularity is between the design and output of software outputs, particularly for big and complex projects. In globally distributed software projects, it is potential to take in a discrepancy

between organizational structure and software architecture which may decrease the overall production. The final aim of software modularization is to increase software quality and productivity. A correctly modularized software is easy to maintain and is of great support to the developer.

Software development work can be distributed using different ways, such as by development phases or by complete modules or subsystems allocated to different development sites. Hence, we necessitate to use whichever gives more advantage. And then the latter can be more advantageous than the former which focuses on the importance of modularization. The software development operation is really difficult and modularization can make it more complicated. Nevertheless, modularization is important because it reduces overhead problem at later levels and it too increases the functionality of each situation and cuts back the demand for interaction with remote sites and also facilitates in naming the process more stable.

Problem distribution should be constructed in a proper manner as it reduces many of the troubles connected with communication between teams or groups. Nevertheless, a convincing solution to this problem cannot be found, and then, it is necessary to present a method for modularization before tasks are allotted to different situations. Thesis presents a method for modularization, which needs to be undertaken before work is distributed in a GSD environment. This approach helps to minimize problems in distributed development and maximizes the benefits. Figure 2.1 shows the GSD.

Figure 2.1: Global Software Development

2.1.2 Design Structure Matrix (DSM)

The creation and advancement of complex items oblige exertion and coordinated effort of many members from diverse foundations, bringing about complex connections among both individuals and activities. A large number of the customary administration devices (PERT, Gantt and CPM routines) [4] does not address issues as a result of these complexities. While the devices include the displaying of successive and parallel operations, they break to address interdependency (input and emphasis), which is a shared inconvenience in complex Product Development (PD) ventures.

Consequently, to uproot these issues, a grid based instrument called Design Structure Matrix (DSM) [5] has created. This system varies from customary Product Development apparatuses in light of the fact that it focuses on speaking to data streams as opposed to work processes. The DSM system is a data exchange display that allows the representation of complex undertaking connections keeping in mind the end goal to locate a legitimate arrangement for the assignments being displayed. The DSM strategy is successful for gathering the most ward segments together which can then be used to shape modules. The DSM strategy likewise helps in simple control and capacity on a PC.

In any case, on that point is a specific type of DSM, **Numerical DSM (NDSM)**, which is proficient to give more itemized data, which renders a more fair comprehension of the plan by exhibiting the adequacy of the association. In any case, the presence of the relationship is insufficient to recognize the most ward parts, NDSM is more attractive than the DSM. Figure 2.2 below shows the Design Structure Matrix.

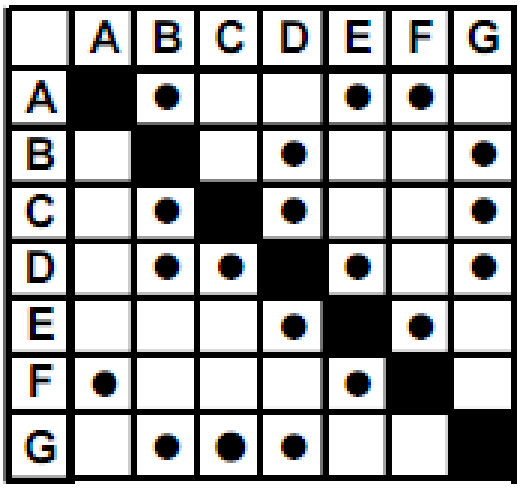

Figure 2.2: Design Structure Matrix

2.1.3 XMI

XMI (XML Metadata Interchange) is a proposed utilization of the Extensible Markup Language (XML) that is planned to give a standard approach to developers and different clients to exchange information about metadata (data about data). In particular, XMI is expected to help programming architects utilizing the Unified Modeling Language (UML) with distinctive dialects and advancement instruments to trade their data models with one another. Furthermore, XMI can likewise be utilized to exchange data about information stockrooms. Adequately, the XMI arrangement institutionalizes how any arrangement of metadata is recognized and obliges clients crosswise over numerous businesses and working situations to see information the same way. Figure 2.3 below shows the xmi format.

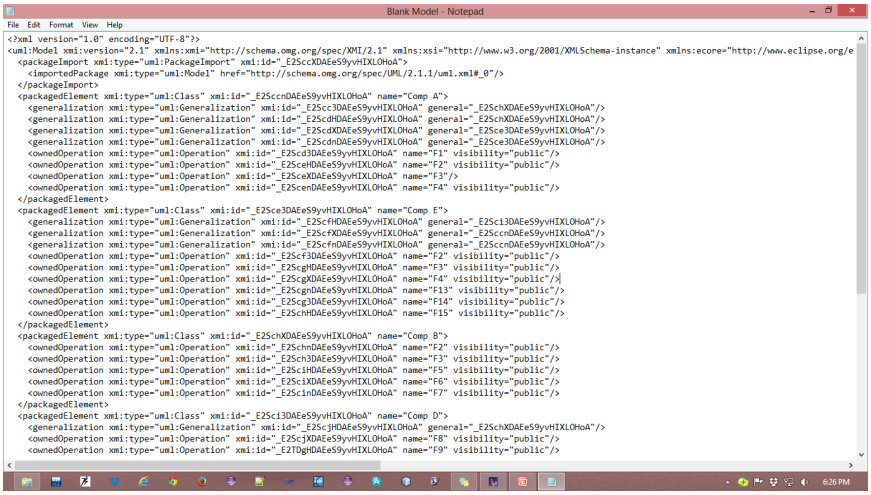

Figure 2.3: XMI

2.1.4 Modular System [8]

A measured framework, MS, is an uncommon instance of a product framework indicated by a chart, S, which has n hubs apportioned into modules, mk, k = 1,$\cdots$,nM. Figure 2.4 below shows the modular system.

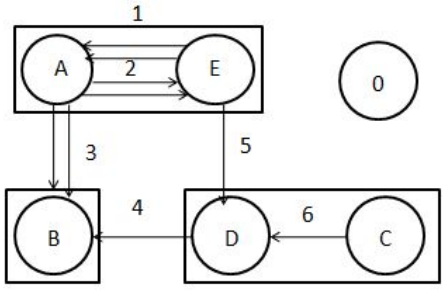

Figure 2.4: Modular System

2.1.5 System Graph

Given a measured framework, MS ,with n nodes parceled into n_m modules, the framework chart, S,has all nodes in MS and every one of its edges, in addition to a disengaged node speaking to the frameworks environment. We are indexing nature

node as i = 0, and the nodes in MS as i= 1 ,$\cdots$, n. The system scope can't avoid being described by the given nodes and edges. We explicitly address the unspecified environment by a single isolates node. For estimation of coupling, we make a further pondering, an intermodule-edges outline, which is a subgraph of S.

2.1.6 Intermodule-edges graph [10]

Given a measured framework, MS, and its structure framework diagram, S, its inter-module edges graph, S^*, embodies all nodes in S and all its intermodule edges. It is repetitive for subgraph S^* to be a related outline. For example, Figure 5.1 depicts the intermodule edges subgraph,S^*, for the deliberate structure in Figure 2.4. For estimation of connection, we make a further pondering, an intramodule edges outline, which is a subgraph of S. The properties of connection focus on intramodule edges.

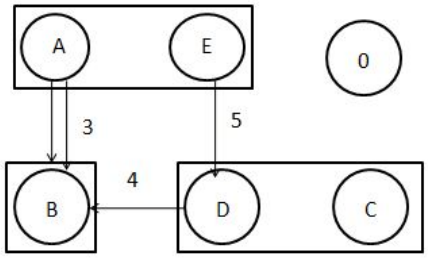

Figure 2.5: Inter-module edges Graph

2.1.7 Intramodule-edges graph [10]

Given a measured framework, MS, and its system diagram, S, having intra-module edges diagram, S_0, which involves all nodes in S and all intramodule edges. Additionally, it is excessive for subgraph S_0 to be an associated graph. For example, Figure 2.6 depicts the intra-module edges subgraph, S_0, for the disengaged system in Figure 2.4.

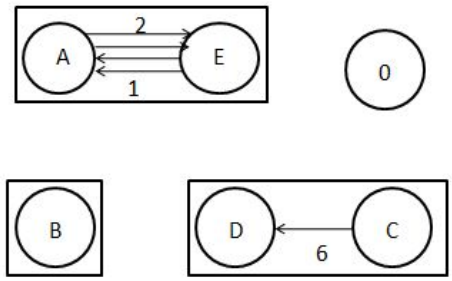

Figure 2.6: Intra-module edges graph

2.1.8 Coupling

In software building, coupling is the style and level of association between software modules, an undertaking of how firmly associated two schedules or modules are the power of the connections between modules.

Coupling is generally appeared differently in relation to cohesion. Low coupling regularly connects with high attachment, and the other way around. Low coupling is regularly a sign of an all around organized PC framework and a decent arrangement, and when blended with high union, bolsters the widespread closures of high intelligibility and viability.

2.1.9 Coupling of a modular system [10]

We are having a particular framework, MS, and also its inter-module edges subgraph, S, the inter-modular coupling of a modular system is the least descriptive length of the relationships in S.

$$\text{Coupling(MS)} = \sum_{i=1}^{n} I(S_i) - I(S)$$

The aggregate sum of data in the graph structure is known as the minimum description length $I(S)$.

$$\text{I(S)} = \sum_{j=0}^{n}(-logP_{L(j)})$$

This equation is used in the above equation for finding the coupling of the modular system.

2.1.10 Intramodule Coupling [10]

The minimum description length of the relationships in $S^{'}$ is known as the intra-module coupling of a particular framework, MS

$$\text{Intramodule Coupling(MS)} = \sum_{i=1}^{n} I(S_i^{'}) - I(S^{'})$$

2.1.11 Cohesion of a modular system [10]

The minimum description length of the relationships in $S^{'}$ divided by the minimum description length of the relationships in $S^{(n)}$ is known as cohesion of a modular system, MS.

$$\text{Cohesion(MS)} = \frac{Intramodule Coupling(MS)}{Intramodule Coupling(MS^{(n)})}$$

where, Intramodule Coupling($MS^{(n)}$) = (n-1)log(n+1) for n>2.

Chapter 3

Literature Review

A method for modularization in Unified Modeling Language Specification for Global Software Development [4] by Dilani Wickramaarachichi and Richard Lai, In ISEC,New Delhi (2013). This paper explains a method for modularization based on formal specification. The proposed method incorporates two different analyses: dependency analysis and functionality analysis. The final decision given by the method is based on the results of both the analyses and priority is given to each analysis depending on the project objective. This framework is designed to suggest an approach for modularization as an assistance for the project manager. The suggestions given by the model is transparent, the project manager is allowed to modify the recommendation at any time. Although the proposed method is mainly for component based GSD. However, the complexity of the interaction and functions demands additional information which may not be available at the early stage of the development. Therefore, according to the information available at the early stage, this method provides an acceptable suggestion which can enhance the success of GSD. Finally, this paper illustrates a formation of modules.

An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method [5] by Yassine and Ali (2004). In this paper the author has provided an introduction about DSM method which is an alternative approach to project management methodology for managing composite PD projects. This approach is helpful in simply inspecting and building the DSM. Even without detailed analysis, building a DSM model of a project, improves visibility and understanding of project/system complexity. The DSM method supports a major need in engineering design management: documenting information that is exchanged. The method provides visually powerful means for capturing, communicating, and organizing engineering design activities and architectural issues such as project team formation and product architecture.

Global software engineering: The future of socio-technical coordination [1] by Herbsleb, Damian(2007). This paper helps us in understanding what global software is. As nowadays more software projects are running in geographically distributed environments. Very often large software projects are developed by multiple teams to meet deadlines and deliver the product on time. In a GSD environment, multiple teams are located geographically dispersed locations and the work distribution process has become an important function in GSD. The work distribution process can assists in benefiting GSD(e.g. availability of people, closeness to the customer) and its risks(e.g. communication problem, Inexperienced workforce) [2]. Therefore work should be distributed after a thorough study on the different tasks and the various locations. To do this, the categorization of work should be undertaken before it is distributed. This is known as modularization. Hence, it generates economic, technical, organizational issues.

Measuring Coupling and Cohesion of Software Modules: An Information Theory Approach [10] by E. B. Allen and T. M. Khoshgoftaar(2001). In this paper, some software metrics are explained based on information theory approach. Coupling and Cohesion of modular system are found out with the help of information each node bears. Usually a graph shows information about the software and how they are related with other components. OOD(Object-Oriented Design) is also based on various graphs. Cohesion and Coupling are attributes that is the outline for the connectivity or degree of interdependence among subsystems and within measures of other attributes, coupling and cohesion helps in the contribution to the prediction of software quality.

Chapter 4

Modularization

4.1 Introduction

Global Software Development(GSD), nowadays, is becoming a standard in software industry and it is getting more difficult to implement these effective strategies. Usually,large software projects are developed by multiple teams to deliver the product on time and meet deadlines. In this environment multiple teams are located in geographically dispersed locations and hence work distribution process becomes an important function in GSD. The work distribution process can assists in benefiting GSD(e.g. availability of people, closeness to the customer) and its risks(e.g. communication problem, Inexperienced workforce) [2].

Work is distributed after a thorough study of different tasks and various locations, so to do this categorization of work should be undertaken before it is distributed. This is said to be as modularization. The plan of modularity is midst the design and production of software artifacts, especially for large and complex projects. In globally distributed software projects,it is possible to have a difference between software architecture and organizational structure which may reduce the overall productivity. The final goal of software modularization is to increase software project quality and productivity. A properly modularized software is very easy to maintain

and is of great support to the developer.

Software development work can be distributed using different ways such as by complete modules or subsystems allocated to different development sites or by development phases. Hence, the one which gives maximum benefit is used. So the former can be more advantageous than the latter which focuses on the importance of modularization. As we know software development process is too difficult and if modularization is added it will makes it way more complicated. Anyways, modularization is much more important because it increases the functionality of each site and it also reduces overhead problem at later stages and also reduces the need for interaction with remote sites and also helps in making the process more stable.

Reduction of many of the problems associated with communication between teams or groups so problem distribution should be made in a proper manner. However, a convincing solution to this problem cannot be found, so, it becomes necessary to introduce a method for modularization before tasks are distributed to different sites. This thesis presents a method for modularization which needs to be undertaken before work is distributed in a GSD environment. This approach helps to minimize problems in distributed development and maximizing its benefits.

4.2 Steps used for modularization

Modularization method used is based on two different ways. They are as:

4.2.1 Dependency Analysis

First one identifies the most dependent components and suggests grouping them together which helps to reduce communication difficulties.

Some steps are followed in this step:

- We analyze the UML Component Based System Specification(CBSS) and represent component dependency in NDSM as shown in fig(2).

	A	B	C	D	E
A					2
B	2			1	
C					
D			1		1
E	2				

Figure 4.1: Componet Interaction Matrix

- After making the component interaction matrix the matrix is rearranged using partition algorithm.

	C	D	E	A	B
C					
D	1		1		
E				2	
A			2		
B		1		2	

Figure 4.2: Rearranged component interaction matrix after partition

- After rearranging the matrix the module structure is suggested based on dependency analysis.

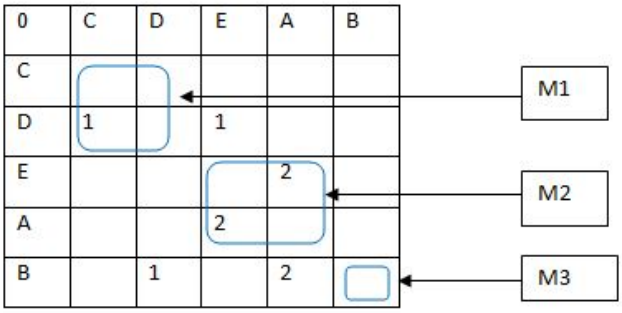

Figure 4.3: Module Formed

The principal step known as **Dependency Analysis**, parses the XMI representation of UML 2.x. IBM Rational Rose Architecture(RSA) and helps us to draw the UML diagram, and afterward XMI is generated, which is utilized as input for future. A parser is suggested which parses the XMI record to concentrate data about messages, no of capacity identified with distinctive parts, no of nodes related, section structure and consolidated part. Messages are recognized that are sent from any non client article to client object. Now utilizing these separated data a framework known as Component Interaction Matrix is created.

Presently, **Partitioning** is the method for controlling the Numerical DSM(NDSM) [5] [11] lines and sections such that the new NDSM course of action brings all the data stream close to the diagonal. Therefore, one can distinguish consecutive, parallel and coupled assignments to shape modules where the more poor agreements are assembled together. Partition algorithms proceed as follows :

4.2.2 Functionality Analysis

Second stride in this methodology known as **Functionality Analysis** considers the errands which have comparative functionalities and recommends gathering them together which helps in lessening the product advancement exertion. The same methodology of parsing is done.After parsing the XMI record extraction of data about messages,no of hubs related,no of capacity identified with distinctive parts,

Algorithm 1 Partition Algorithm

Input: Component Interaction Matrix

Output: Rearranged Matrix

1: The element without input are identified first.

2: An empty row in the NDSM helps in identification of element without input.

3: Now placing these elements in the top of the NDSM.
Once an element is rearranged, it is removed from the NDSM.
and this very step is repeated on the remaining elements.

4: Element that delivers no information to other element in matrix is identified.
These can be easily identified by an empty column in the NDSM.

5: Now placing these elements in the bottom of the NDSM.
Once an element is rearranged, it is removed from the NDSM
and step is repeated on the remaining elements.

6: All the above steps are repeated until the matrix is sorted.

structure of sections and consolidated pieces is finished. Messages that are sent from any non client article to client item are identified.Using all the separated data a network known as Functionality grid is produced. Before discovering usefulness lattice we discovered a table known as capacities identified with every assignment which helps in shaping the usefulness network.

A	B	C	D	E
F1	F2	F1	F8	F2
F2	F3	F2	F9	F3
F3	F5	F6	F10	F4
F4	F6	F8	F11	F13
	F7	F9	F12	F14
				F15

Figure 4.4: Functions related to each task

	A	B	C	D	E
F1	1	0	1	0	0
F2	1	1	1	0	1
F3	1	1	0	0	1
F4	1	0	0	0	1
F5	0	1	0	0	0
F6	0	1	1	0	0
F7	0	1	0	0	0
F8	0	0	1	1	0
F9	0	0	1	1	0
F10	0	0	0	1	0
F11	0	0	0	1	0
F12	0	0	0	1	0
F13	0	0	0	0	1
F14	0	0	0	0	1
F15	0	0	0	0	1

Figure 4.5: Functionality Matrix

	A	B	C	D	E
A		2	2		3
B			2		2
C				2	1
D					
E					

Figure 4.6: Similar functions between tasks

After finding the functionality matrix we suggest the module structure on the basis tasks having similar functionalities are developed in one location, development effort is minimized. Further, it increases the maintainability of the system. As show in Figure 4.6, the functionality analysis suggests that tasks A & E should be modularized as they have maximum common functions. The remaining tasks B-C and tasks C-D both have 2 similar functionalities. Now we need to analyze the modules and form the best module. We take different aspects to form the best module such as to minimize communication and coordination problems, we get the module structure as:

M1:A-E

M2:C-D

M3:B

After these analysis we finally decide for final modularization which is the main objective of first part of our project. So according to the suggestions found out of the dependency analysis part and the functionality analysis part, final decision to form modules is as follows :

M1:A-E

M2:C-D

M3:B

4.3 Summary

This chapter helps in forming the best module of the GSD using modularization. Two different analysis is used for the formation of the best modules. The Dependency analysis is the first analysis which takes help of dependency between modules to find the best modules. The functionality analysis is the second analysis which takes help of functions related to each other and helps in the formation of the best module. After these two analysis the best module is found out of the two analysis and the final module structure is formed.

Chapter 5

Finding Coupling & Cohesion

5.1 Coupling

In software engineering, coupling is the trend and level of association between software modules, a measure of how firmly associated two schedules or modules are the strength of the connections between modules.

Low coupling always connects with high attachment, and the other way round. Low coupling is always a sign of an all around organized PC framework and a decent arrangement, and when combined with high cohesion, then it gives high practicality and clarity.

5.1.1 Types of coupling

Common coupling

Global coupling (otherwise called Common coupling) happens when two modules have the same worldwide information. When the correlative things are changed all the modules also get the same changes.

Content coupling

Content coupling happens when a module adjusts or depends on the interior activity of all the other module. So, changing the manner in which the second module produces information (area, sort, timing) will provoke changing the dependent module.

Procedural programming :

A module here suggests to a subroutine of any sort, i.e. an arrangement of one or more explanations which have a name and their own arrangement of variable names.

External coupling

External coupling happens when two modules share a remotely forced correspondence convention, information design, or gadget interface. This is fundamentally identified with the correspondence to outside gadgets and devices.

Stamp coupling

Stamp coupling also known as Data-structured coupling happens when modules share a complex information structure and utilizes only a piece of it, conceivably an alternate port (e.g., running an entire disc to a capacity that only calls for one field of it). This may prompt changing the way a module peruses a record in light of the fact that a field that the module does not need has been commuted.

Control coupling

Control coupling is one module controlling the stream of another, by passing it data on what to do (e.g., passing a what-to-do flag).

Message coupling

The easiest and loosest kind of coupling. Message or parameters passing helps in expertizing the state decentralization and segment correspondence.

Data coupling

Parameters and many other things help a module to offer information in data coupling. Data is the elementary part and it is the main information shared.

No coupling

Modules do not interact at all with each other.

5.1.2 Properties of coupling of a module

Table 5.1: Properties of coupling of a module

Sl. No.	Properties	Description
1.	Nonnegativity	It is nonnegative.
2.	Null value	It is null if its set of inter-module edges is empty.
3.	Monotonicity	Including an inter-module edge to a module does not lessening its module coupling.
4.	Merging of modules	If two modules are merged to form a new module then the module coupling of the newly formed is not greater than the sum of the module couplings.
5.	Disjoint module additivity	If two modules which have no inter-module edges between nodes are merged to form a new module then the coupling of the modular system is equal to the couplings of modules.

5.1.3 Inter-modular Coupling [10]

In inter-modular coupling, we first examine the subgraph which have all the nodes in the modular system and here the environment node is also present [5]. The inter-module edges are present in the other modules. we number the environment node as i = 0, and the rest of the node as i=1,$\cdots$,n. Subgraph should be a always not be a connected graph. Figure 5.1 below shows the inter-module edges subgraph for the modular system in Figure 2.4.

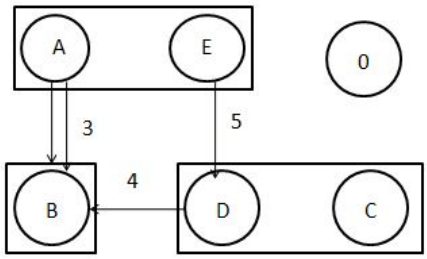

Figure 5.1: Inter-modular edges Graph

Now we label each node lying on the edges, this will help us when we will analyze the pattern in future. The graph is represented using a node $\times$ edges table where all the cell gives us information about which node is attached with which edges. Now this node logo becomes a binary patten of the values in the table.Table 5.5 shows nodes $\times$ edges table of S.

Table 5.2: Example Intermodule Edges Graph

Module	Node	3	Edges 4	5	$P_L(i)$
Environ.	0	0	0	0	0.334
m_1	A=1	1	0		0.167
	E=5	0	0	1	0.167
m_2	C=3	0	0	0	0.334
	D=4	0	1	1	0.167
m_3	B=2	1	1	0	0.167

$P_L(i)$ is the piece of that row's pattern out of fifteen rows. the total amount of information in the graph structure is known as the minimum description length, $I(S)$.

$$Inter - moduleCoupling = \sum_{i=1}^{n} I(S_i) - I(S) \tag{5.1}$$

where,

$$I(S) = \sum_{j=0}^{n} (-logP_{L(j)}) \tag{5.2}$$

An environment node is a disconnected node and a disconnected node has no subgraph S_i, hence, I(S_0) = 0, for environment node.

5.1.4 Coupling of a modular system [10]

The minimum description length of the relationship in S is known as the coupling of the modular system.

$$Coupling(MS) = \sum_{i=1}^{n} I(S_i) - I(S) \tag{5.3}$$

The total amount of information in the graph structure is known as the minimum description length $I(S)$.

Equation 5.2 is used in equation 5.3 for finding the inter-modular coupling of a modular system.

5.1.5 Intra-modular Coupling [10]

We take a measured structure, MS, which is represented with the help of a diagram with n nodes distributed into modules, and the surroundings is always represented by separated node, taking the subgraph , S', comprising of all nodes in MS in addition to environment node, and all intra-module edges. The end point of an intra-module edge are not in other module but only present in the same module. We number the environment node as i = 0 and the other nodes as $i = 1, \cdots, n$. We use the table for node and edges.

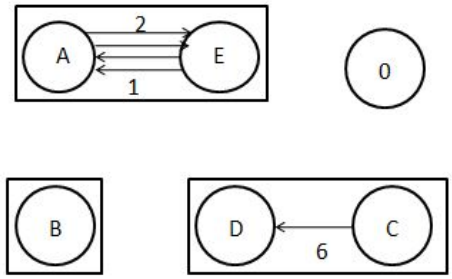

Figure 5.2: Intra-module edges graph

Table 5.3: Example Intra-module Edges Graph

			Edges		
Module	Node	1	2	6	$P_L(i)$
Environ.	0	0	0	0	0.34
m_1	A=1	1	1	0	0.34
	E=5	1	1	0	0.34
m_2	C=3	0	0	1	0.34
	D=4	0	0	1	0.34
m_3	B=2	0	0	0	0.34

Figure 5.2 shows the intra-modular edges subgraph for the modular system in Figure 2.4, and Table 5.3 shows its *nodes* $\times$ *edges* containing $P_L(i)$ which helps in finding cohesion and coupling.

The minimum description length of the relationships in S' is known as the the intra-modular coupling of a modular system, MS.

$$IntramoduleCoupling(MS) = \sum_{i=1}^{n} I(S_i^{'}) - I(S^{'}) \tag{5.4}$$

By equation in inter-module coupling, this is equivalent to the excess entropy of S' times the number of nodes, $n + 1$.

5.2 Cohesion

Cohesion suggest to what level the components of a module have a place together. It tells us how strongly related every bit of information is, by the modules of a software module is.

It is an ordinary kind of estimation and is regularly shown as "low coupling" or "high cohesion". Modules having high attachment is the best which can be explained as high cohesion is associated with a couple of appealing qualities of

software, including understandability, re-usability, reliability and power. However, low cohesion is involved with unacceptable qualities, like, being difficult to care for reuse, test and even read.

Cohesion is always related with coupling, a substitute thought. High cohesion routinely relates with loose coupling, and the other way around.

5.2.1 Types of cohesion

Cohesion have these types from the most exceedingly repulsive to the best, are as per the following:

Coincidental cohesion

When some parts of a module are collected in a manner then they are known as coincidental cohesion. The basic relationship between the The parts have to be assembled, this is the basic relationship between them (e.g. an "Utilities" class).

Logical cohesion

When some parts of a module are joined because they are differentiated logically to do same, maybe that they are different by nature (e.g. gathering all mouse and console info handling together) is known as logical cohesion.

Temporal cohesion

When parts of a module are joined when they are arranged - the parts are seen only for some particular time during program execution is known as Temporal cohesion.

Procedural cohesion

When parts of a module are joined having knowledge that they for most part take after a particular progression of execution is known as Procedural cohesion.

Communicational cohesion

When parts of a module are joined having knowledge that they deal with the same data (e.g. a module which takes a shot at the same record of information) then it is known as Communicational cohesion.

Sequential cohesion

When parts of a module are accumulated in light of the way that the yield beginning with one area is the information then onto the following part like a consecutive development framework is known as Sequential cohesion.

Functional cohesion

When parts of a module are assembled in light of the way that they all add to a single all around undertaking of the module is known as Functional cohesion.

5.2.2 Properties of cohesion of a module

Table 5.4: Properties of cohesion of module

Sl. No.	Properties	Description
1.	Nonnegativity and Normalization	It belongs within a specified interval. Cohesion(MS) ϵ [0,Max].
2.	Null value	If the set of intra-module edges is empty then it is null.
3.	Monotonicity	The cohesion of module does not decreases, whenever we add an intra-module edge to the modular system.
4.	Merging of modules	The cohesion of the modular system is not greater than the cohesion of the modular system, if two unrelated modules, are merged to form a new module.

5.2.3 Cohesion of a modular system [10]

Intra-module coupling, talked about above in this section, measures data on intra-module connections. Cohesion and intra-module coupling are measures of different types of relationships within modules; intra-module coupling is a quantity of bits, yet cohesion is a standardized fraction.

We are having a measured framework, MS, having n number of nodes, the intra-module edges subgraph, S', is indicated as above. Now considering the framework $MS^{(n)}$, comprises of a complete graph of n nodes in one module, where each node is joined with the other node. Basically, $MS^{(n)}$ is the "most cohesive" framework comprised with n number of nodes.

Similar to the coupling measures, we use $nodes \times edges$ tables as a convenient

representation for S' and $S^{(n)}$.

The minimum description length of the relationships in $S\prime$ divided by the minimum description length of the relationships in $S_{(n)}$ is known as the cohesion of a modular system, MS.

$$\text{Cohesion(MS)}=\frac{IntramoduleCoupling(MS)}{IntramoduleCoupling(MS^{(n)})}$$

Cohesion of MS is equivalent to the ratio of excess entropies, $\frac{C(S\prime)}{C(S_{(n)})}$, because S_n and $S\prime$ both of them are having same no of nodes.

5.2.4 Results of cohesion and coupling of modular systems

Table 5.5: Results for coupling and cohesion

Case Study	Coupling(in bits)	Cohesion
ATM System	0.616	1.66
Library Information System	43.26	5.797

5.2.5 Summary

This chapter deals with modular system and helps in finding cohesion and coupling of the modular system using the help of Information theory Approach.

Chapter 6

Future Work & Conclusion

6.1 Conclusion

This thesis presents a method for modularization based on formal software application. The proposed method incorporates two different analyses: dependency analysis and functionality analysis. The decision given by the method is based on the results of both the analyses depending on the project objective. The results of both the analyses is considered and the the best result of both the module is found out and the best module is formed. In case the result of both the analyses both differ a lot then the project manager has the right to make the modules.

After analyzing inter component dependency and common functionalities in each component, suitable combinations is selected for modularization to achieve the project objective. Suggestion for formation of module arising from the dependency analysis helps to reduce communication and coordination costs whereas suggestion arising for module formation from the functionality analysis helps to reduce the development effort. Decision regarding final modularization is based on the best way to achieve the main objective of the project. Further, the nature of the project and the size of the functions will be taken into consideration to decide the most cost effective combinations for module structures.

After finding the best modules we then find the coupling and cohesion of the modular system. Cohesion and coupling is qualities which outline the degree of inter-dependence inside of subsystems and among subsystems, separately. At the point when utilized as a part of conjunction with measures of different properties, cohesion and coupling can add to an evaluation or forecast of software quality.

6.2 Future Work

- In future this whole process can be applied for all the other UML diagrams and comparison can be made for finding the best module structure.
- In future CK-Metrics of the modular system can be found out for getting a quality software.

Bibliography

[1] Damian, D. and Moitra, D.,(2006), Gobal Software Development:How Far Have We Come?, Software, IEEE, vol.23, pp. 17-19.

[2] Lamersdorf, A.,Munch,J.,and Rombach,D.,(2009), A survey on the state of the practice in distributed software development: criteria for task allocation. In Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on, 2009, pp. 41-50.

[3] Cai, Y.(2009),Assessing the Effectiveness of Software Modularization Techniques through the Dynamics of Software Evolution. In Contemporary Modularization Techniques(ACoM.08), pp.40

[4] Dilani Wickramaarachichi and Richard Lai, A Method for modularization in Unified Modeling Language Specification for Global Software Development,In India Software Engineering Conference,New Delhi (2013).

[5] Yassine, Ali, "An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method",Quaderni di Management,it, No.9,2004.

[6] Herbsleb, J. (2007), Global software engineering: The future of socio-technical coordination, in 2007 Future of Software Engineering, 2007 , 00.188-198.

[7] E. B. Allen and T. M. Khoshgoftaar. Measuring coupling and cohesion: An information-theory approach. In Proceedings of the Sixth International Software Metrics Symposium, pages 119-127, Boca Raton, Florida USA, Nov. 1999. IEEE Computer Society.

[8] L. C. Briand, S. Morsaca, and V. R. Basili, Property based software engineering measurement. IEEE Transaction on software Engineering,22(1):68-85,Jan. 1996.

[9] Khare, A. and Bangare, M.(2011), Measuring The Quality Of Object-Oriented Software Modularization. In International Journal on Computer Science and Engineering(IJCSE),vol.3.

[10] E. B. Allen and T. M. Khoshgoftar, Measuring Coupling and Cohesion: An Information Theory Approach. In proceeding of the Sixth International Software Metrics Symposium, pages 119-127, Boca Raton, Florida USA, Nov 1999. IEEE Computer Society.

[11] Eppinger, S., Whitney, D., Smith, R., and Gebala, D.(1991), Organizing the tasks in complex design projects, In Computer-Aided Cooperative Product Development,pp. 229-252,1991.

[12] E. B. Allen. Information Theory and Software Measurement. Phd thesis, Florida Atlantic University, Boca Raton, Fl, Aug. 1995. Advised by taghi M. Khoshgoftaar.

[13] Mall R. 2009. Fundamentals of software engineering(4th edn.). PHI learning Pvt. Ltd.

[14] M.H. van Emden. Hierarchical decomposition of complexitu. Machine Intelligence,5:361-380, 1970.

[15] DeLone, W., Espinosa, J., Lee, G. and Carmel, E. (2005), Bridging global boundaries for IS project success. In Proceedings of the 38th Hawaii International Conference on System Sciences.

[16] Narduzzo, A. and Rossi, A., (2003) Modular design and the development of complex artifacts:Lessons fron free/open source softwares,Quaderno,DISA,vol.78.

[17] Salger, F. (2009), Software architecture evaluation in global software development projects, in On the Move to Meaningful Internet Systems: OTM 2009 Workshops, 2009, pp. 391-400.

[18] hoffman, D., and Weiss, D. (2001), Software fundamentals: collected papers by David L. Parnas, ed: Addison- Wesley,2001.

[19] Card, D., Page,G. and McGarry,F.(1985),Criteria for software modularization, 1985, pp. 372-377.

[20] Xiao, R. and Chen, T. (2010), Research on design structure matrix and its applications in product development and innovations: an overview, International Journal of Computer Applications in Technology, vol. 37, pp. 218-229,2010.